1

Basic Installation

Ever wondered where Basic came from? Much maligned but still the most popular programming language in the world, the Basic language has enjoyed a colorful past and many evolutions to get where it is today. In this chapter, you'll discover the origins of this powerful programming language. You'll install Visual Basic 2005 Express Edition along with the supporting applications and frameworks you'll need to write programs effectively. And finally, yes, you will indeed write your first program.

By the time you hit the end of this chapter, you'll be familiar with how Visual Basic is put together and be ready to create programming projects that will form the basis for all of your solutions from here on out.

In this chapter, you learn about the following:

- The history of Visual Basic as a language
- Installing Visual Basic Express and its dependencies
- Creating your first program

Where Did Basic Come From?

If you tell friends or work colleagues who are experienced in programming that you're going to learn Visual Basic, there is a good chance that they'll look at you with a question in their eyes. That questioning glare is usually an indicator that they're in what I call the "other half" of the programming world. This is the group of programmers who still believe that Basic is not a real programming language, and should be reserved for people who don't know how to write a "real" program.

If this happens to you, just look at them and smile. For while Basic has indeed had a rocky history, the last couple of versions of Visual Basic rival the best alternatives in development, and with Visual Basic 2005 Express's extra features that make it even easier to create full-blown solutions, not only will your programs be able to achieve the same results as the best professional coder, but you will also be able to do it in less time — much less time.

However, to be fair, this section provides a quick rundown of where Visual Basic Express came from, just so you know how far it has come. You'll learn that Visual Basic has a rich past that has helped it evolve into a solid, respected language that often leaves the more recent programming languages scrambling for a foundation on which they can be compared against it.

The Basic programming language was first created back in 1964 — more than 40 years ago. Its very inception was meant to make programming easy and more accessible. In fact, the name was actually originally an acronym that stood for Beginner's All-Purpose Symbolic Instruction Code. It was designed as an interim step for students when they were learning programming concepts for more complex languages such as Fortran.

In the 1970s, Bill Gates and Paul Allen got involved and worked with the company MITS (Micro Instrumentation Telemetry Systems) to develop a version of Basic for the Altair PC. From that humble beginning, Gates and company ported Basic to various other computing platforms, and by the end of that decade, most computers had some form of the Basic language. It was from this starting point that both its ease of use and popularity, as well as the disparaging opinions of many hardcore programmers, sprang.

When DOS was first released for the early PCs, versions of a Basic interpreter were distributed along with the operating system. Programming code can be executed in two ways — interpreted or compiled:

- When it is **compiled**, it is assembled into the underlying machine code and can execute fast. However, the compilation can take a while, and the program will not execute at all if even one error is present.
- An **interpreter**, on the other hand, requires another program to run through the code one line at a time and execute it piece by piece. While this is slower than compiled code, it doesn't require a compilation routine before running, and it can execute working code up to the point where an error occurs. Basic, and Visual Basic in particular, requires some form of a runtime component because of the interpretive nature of the language compilers.

Microsoft took the command-line interpreter to the next step and introduced QuickBasic. QuickBasic did actually compile the code into an executable, but it was still slow in comparison to the professional languages on the market. In the late 1980s and early 1990s, Alan Cooper created a prototype that enabled a developer to dynamically add components, then called *widgets,* to a program running off a small, custom-built language engine. Microsoft bought the concept and combined it with QuickBasic to form Visual Basic 1.

And Then Came Visual Basic

Visual Basic was a revolution to Basic programmers worldwide as it enabled them to drag and drop controls from a toolbox onto their forms without having to write any code at all. It also changed the focus of the actual code to an event-oriented model that reacted to things happening, as opposed to making things happen.

Visual Basic's versatility enabled third-party companies to develop add-ins and additional controls that Visual Basic programmers could use in their own applications, and the popularity of the language grew hugely.

Subsequent versions of Visual Basic introduced database support (ODBC in VB2, and Jet in VB3) and the ability to create your own add-ins and classes (in VB4), and ultimately your own controls (in VB6). While all of this was happening, Basic appeared in other applications such as Access Basic and VBScript for Internet Explorer. This integration of Basic as a way of programmatically accessing features in Windows and applications culminated in Visual Basic for Applications, which first appeared in Microsoft Office 97.

Throughout all these stages of its evolution, however, Visual Basic was still crippled with additional runtime components and a (much) less than perfect implementation of object-oriented programming that hurt its reputation in the performance and pure programming stakes.

That all changed with .NET. Visual Basic .NET was the first fully compiled language and required no extra runtime component other than the one required by all other .NET languages — the .NET Common Language Runtime (CLR). Visual Basic .NET programs compile down to the same assembled code that the other .NET languages do; and because of this, Visual Basic has no performance issues in comparison to C# or C++.

> **In the last few paragraphs, several programming terms have been used that you may not be familiar with. If you are new to programming, then the next few chapters will be extremely useful to you — particularly the information in Chapter 2 that explains the most commonly used object-oriented programming terms that you'll encounter in Visual Basic Express.**

The Old and the New

The beauty of this latest move for Basic is that it has not lost the ease of use and additional features that make it the choice of many programmers — wizards, intuitive user interface design, and some excellent debugging features (although edit-and-continue was removed in the early days of .NET, it lives again in Visual Basic 2005 Express).

In fact, the modern development environment for .NET has more in common with the way Visual Basic 6 worked than the C++ equivalent. The toolbox, Solution Explorer, and properties pages are almost unchanged, and the way of associating code with user interface elements is identical to previous versions. For people with previous experience in Visual Basic programming, the only real hurdle is learning how to handle the new way of actually coding — proper object-oriented programming is admittedly different from the way VB6 did it.

So here we are, with a programming language that has evolved over more than 40 years and through many iterations and somehow has maintained a freshness with each release that has kept programmers faithful to it over all that time. It is a language that possesses an incredibly robust and intuitive framework of objects and programming constructs that ease you, as a programmer, into creating full-blown applications almost without thought, and an environment that can produce applications that rival the professionally built solutions on the market in performance and user interface. Visual Basic 2005 Express — want to use it? Thought so.

Let's Get Started

Obviously, before you do anything else, you're going to need to install Visual Basic Express on your computer. Microsoft has fine-tuned this process over the years, and you'll find the steps to be as easy as 1-2-3.

When you go through the Visual Basic 2005 Express Setup wizard, you need to select only a couple of options before the setup process takes over and does the rest for you. After you read and accept the license agreement, the installation program will examine your system and present you with a list of two optional products (see Figure 1-1).

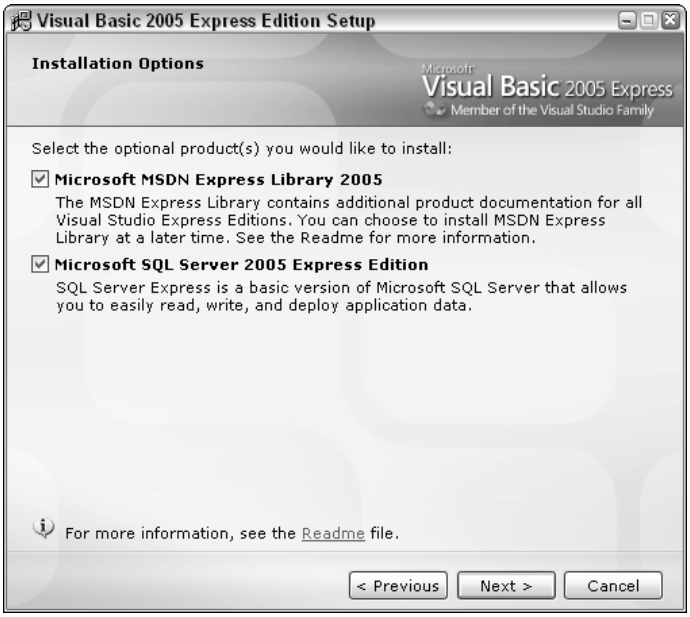

Figure 1-1

As far as I'm concerned, both of the optional components are essential:

- The **MSDN Express Library** includes the documentation for both the .NET Framework and Visual Basic 2005 Express. If you do not include this in the installation, you will have only very rudimentary help available to you without going to the Internet.
- The second option includes **SQL Server 2005 Express Edition** in the installation process so that you can develop full-blown database-based applications. And if you're still not convinced, you will need it to be installed if you want to complete all of the exercises and tasks set out in this book.

> The only exception would be if you already have SQL Server installed on your system. In that case, you could use the existing installation for any database server examples instead, although I cannot guarantee they will work as expected if you are using an older version of SQL Server.

The only other decision that you have to make is to where to install the application. Note that you don't actually get to choose the location of the optional components or the underlying .NET Framework. In addition, this location does not affect the location of the projects you will create — you'll set that location later in this chapter. Once you've made that decision, click Next to start the actual file copy and registration process.

As the installer copies each component over to your computer, it will mark the status on an interim screen. The obvious icons will point out any errors, but most likely you'll encounter nothing but success. In the event of an error, the installation process will advise you as to what steps to take to rectify it before you try again.

Fortunately for you, the rest of the installation is automatic, and while it can take quite some time, you can sit back and have a coffee (and perhaps a Danish) while you wait. When you're presented with the final screen, you have the capability to submit to Microsoft a copy of the installation log so they can check it against what they expect.

> *While many people believe submitting this information is either pointless or a way for Microsoft to gain access to private data, Microsoft does actually find the information useful in fine-tuning its processes, and anything that improves the speed and efficiency of an installation process is something I am 100 percent behind.*

What It Looks Like

Once you have successfully installed Visual Basic 2005 Express, you can start it up by selecting it from your Start menu. Click Start ➪ All Programs ➪ Visual Basic 2005 Express Edition. After the obligatory splash screen identifying the application and version, you'll be presented with an interface much like the one shown in Figure 1-2.

The main program is known as an *Integrated Development Environment*, or *IDE* for short. The IDE of Visual Basic Express has been formed from the experiences of many programmers and many other environments, but it will definitely be familiar to anyone who has programmed in Visual Basic before.

To explore the main elements, you should expand and pin several windows and explorers. As you can see in Figure 1-2, to the right of the Welcome page is an area entitled Solution Explorer. In the top-right corner of this area are three small buttons. The middle one is the *pin*, or auto hide, button. When clicked, this tells the IDE to always show the area, or to automatically hide it when it is not needed. Another window that is currently hidden is the toolbox to the left of the Welcome page.

To better describe the environment, and to start setting it up in a way that will be useful to following the examples in this book, move your mouse over the Toolbox tab on the left, and when the IDE automatically expands it, click the pin button to keep it from automatically hiding. Next, create a basic project by clicking on the File menu, selecting New Project, and clicking OK when the New Project dialog window is shown. This will create an empty form and show it in Design view.

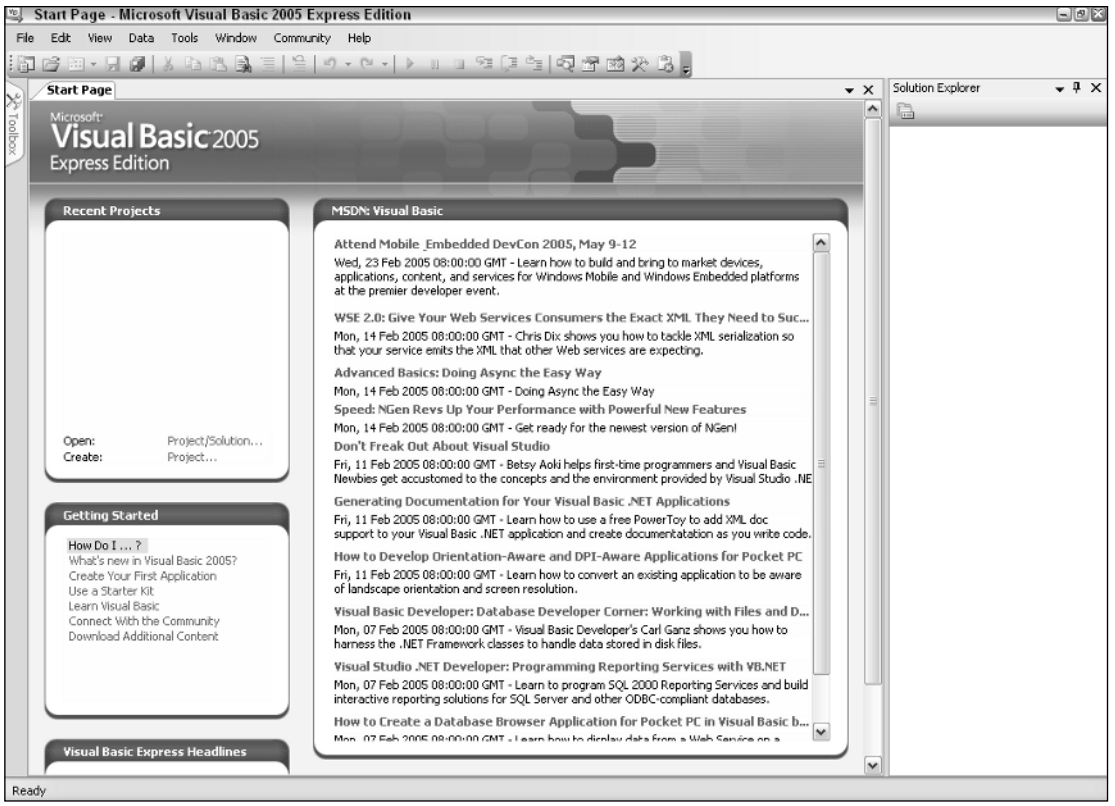

Figure 1-2

To finish setting the scene, double-click on the word Button in the Toolbox window, and the Visual Basic Express IDE will automatically place a button on the form in a default location and with default settings. After you've done all this, the IDE should look like Figure 1-3.

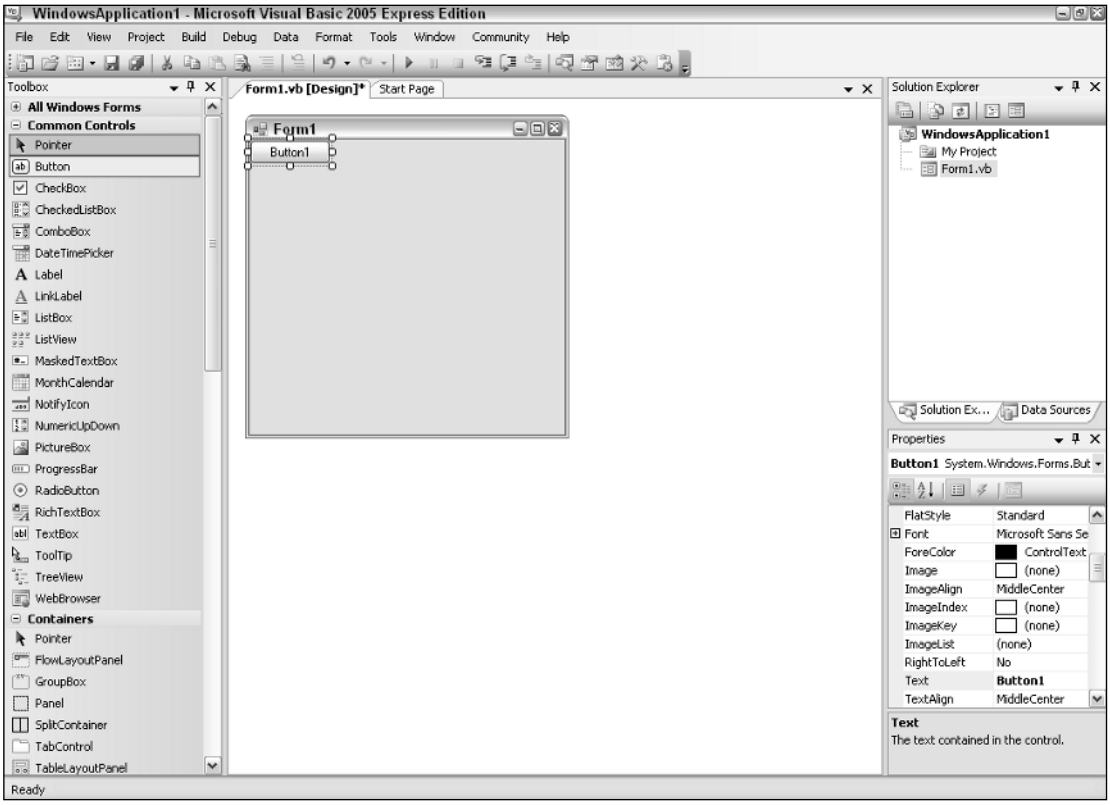

Figure 1-3

The Major Components

Now that the main components of the IDE are visible (and you have even used some!) it's time to tell you what each section does. You should already be familiar with menus and toolbars — they're present in almost every current application. The thing to be aware of in Visual Basic Express is that they're dynamic, and show only the commands that are appropriate to the current context. For example, the Format menu will disappear when you're in a code window, as it doesn't make sense for it to be present when you're writing code. Similarly, the Text Window toolbar won't show when you're designing a form layout.

The next major window is the *Toolbox*. In the next several chapters, you'll use the Toolbox to add various components to your applications, which should give you an idea of what each one does. Every fundamental component you can add to your solution can be found in the Toolbox. To add one to your form layout, you can double-click on it or click-and-drag it to the form.

The objects are grouped into logical sections based on function. By default, you'll find the *Windows Forms* section is expanded and contains many commonly used elements such as buttons and text areas. The other readily available groups deal with data-related components, such as database connections and system components, that give you access to system-level features such as performance monitors and hardware devices.

Moving over to the other side of the main window, you'll find two more essential windows: the *Solution Explorer* and the *Properties window* (both of which are shown in Figure 1-4).

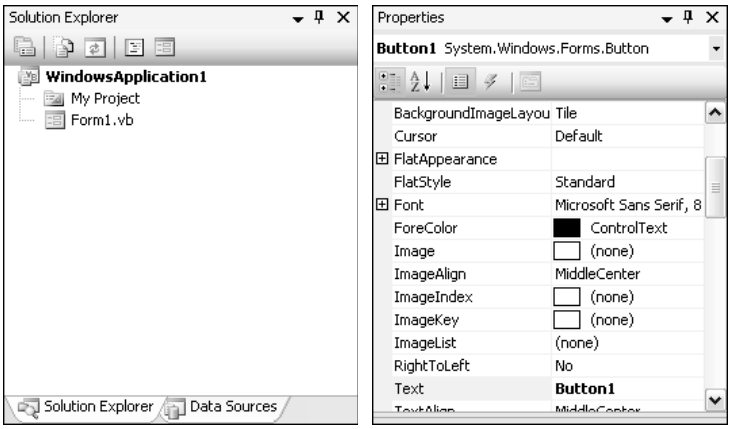

Figure 1-4

- The **Solution Explorer** provides you with a way to navigate through your program's structure, with entries for each form, module, and class, along with supporting files such as the application configuration file. The view is structured in a way similar to Windows Explorer, so you should have no problem navigating your way through the program.
- The **Properties window** gives you access to the various configurable options available to the currently selected item. This can be a form, a server component, or an individual object (such as the Button object shown in Figure 1-4). By default, the Properties window is organized into categories, but you can click the A–Z button to sort the properties alphabetically instead.

The last major areas to cover are the *Error List* and *Task List* windows at the bottom of the IDE. These two windows will not appear until you have compiled or run an application, but after that point, they will always be present by default:

- The **Error List** will be populated with any potential issues with the code and form design of your application. The issues will be broken down into three categories — errors that will stop the program from compiling at all, warnings that indicate a probable runtime error that ought to be investigated before running your program, and informational messages that are purely there for your reference and won't affect the way the program runs.
- The **Task List** contains automatically generated tasks, although you can also manually create your own user tasks. You can use this list to keep an eye on what needs to be done, and you can check individual tasks off as you complete them.

Your First Program

You're actually well on the way to creating your first program in Visual Basic 2005 Express. Earlier in the chapter, you created a Windows Application that generated a blank form. On the form, you added a button. To finish the job, you'll need to write a single line of code that will be executed when a user clicks on the button. The following Try It Out walks you through the entire process of creating the project, adding the button to the form, and writing your first line of code.

Try It Out — Creating Your First Program

If you didn't create the project in the previous part of this chapter, follow these steps:

1. Start Visual Basic 2005 Express. As mentioned previously, you'll find the link to Visual Basic in your main All Programs list on the Start menu.
2. Create a new Visual Basic project by selecting File ➪ New Project. This will present you with the New Project window, listing all available project templates (see Figure 1-5).

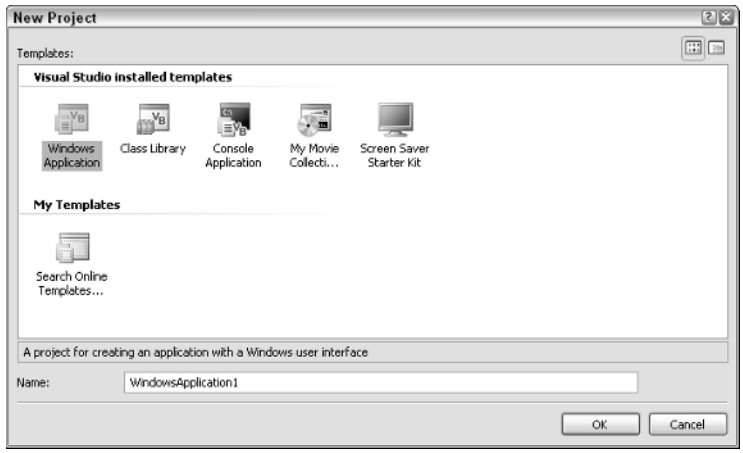

Figure 1-5

By default, Windows Application should be selected. This will create a normal program that runs in Windows. Click OK when you're ready to have the project generated for you.

3. After a moment, you will be presented with a blank form in the center of the IDE. Find the `Button` control in the Toolbox and double-click it to automatically add it to the form in the top-left corner.
4. Select the `Button` object that was added to the form by clicking it once. Locate the `Text` property in the Properties window (it may be easier if you sort it alphabetically) and change it to `Say Hello`. To do this, you should click the right-hand column next to `Text` to access the value (by default it says `Button1`, which is the name of the control).

5. Double-click the button on the form and the IDE will automatically open the code window for this form. It will then create the necessary code to execute your code when the button is clicked, like so:

```
Private Sub Button1_Click(ByVal sender As System.Object, _
    ByVal e As System.EventArgs) Handles Button1.Click

End Sub
```

6. In between the `Private Sub` and `End Sub` lines, write the code `MessageBox.Show("Hello World!")` so that the program appears like this:

```
Private Sub Button1_Click(ByVal sender As System.Object, _
    ByVal e As System.EventArgs) Handles Button1.Click
        MessageBox.Show("Hello World!")
End Sub
```

7. Now you can run your program. The easiest way is to simply press the F5 button, but if you find the menus and toolbars easier to use, you'll find the Start command in the Debug menu. Either way, when you run your program, you'll be presented with a simple form with a single button on it. If you click the button, it will display a message box with the words "Hello World!" (as shown in Figure 1-6). Congratulations — you've written your first complete Visual Basic Express program!

Figure 1-6

What you've done is create a Windows Application — a program designed to run on Windows with a base form. You then added a button to it and wrote actual code to generate a message dialog box when the user clicked it.

That Was Too Easy

Yes, I know — that first program seemed a little too easy, didn't it? That you needed to write only one line of code to actually create a program containing a button on a form that produces a message might seem a little crazy, but that's what Visual Basic Express is all about — making life as a programmer incredibly simple.

To show you that this simplicity extends well beyond the age-old Hello World program, I can show you how to create a simple web browser. The intention is to create a form that has a button, a text input area, and a fully functional web browser on it. When the user clicks the button, the web browser will attempt to navigate to the URL entered in the text area.

Try It Out — Your Very Own Web Browser

1. Start a new Windows Application project in the same way you did in the previous Try It Out exercise.

2. Once the blank form is generated, you need to add a `Button` control, along with a `TextBox` control, and a `WebBrowser`. Because you want to be able to see what's on the web page, resize the form to 500 pixels wide by 460 pixels high. To do this, you can select the form in the design window and click and drag the bottom right corner to the desired size, or you can locate the `Size` values in the Properties window. You'll learn more about properties in more detail in subsequent chapters, but for now overwrite the current setting with the value 500, 460.

3. Once the form size is set, click and drag the three controls from the Toolbox onto the form and then resize them — again using either the click-and-drag method or setting the values directly in the Properties window — so they are laid out as shown in Figure 1-7. You should also set the `Text` property of the button control to the word `Go`.

   You'll notice that as you click and drag controls to move them about or to resize them, small helper lines appear. These lines indicate ideal proximity to the edges of the form or to other controls. In some cases, you'll also see small blue alignment lines that make aligning controls with each other easy.

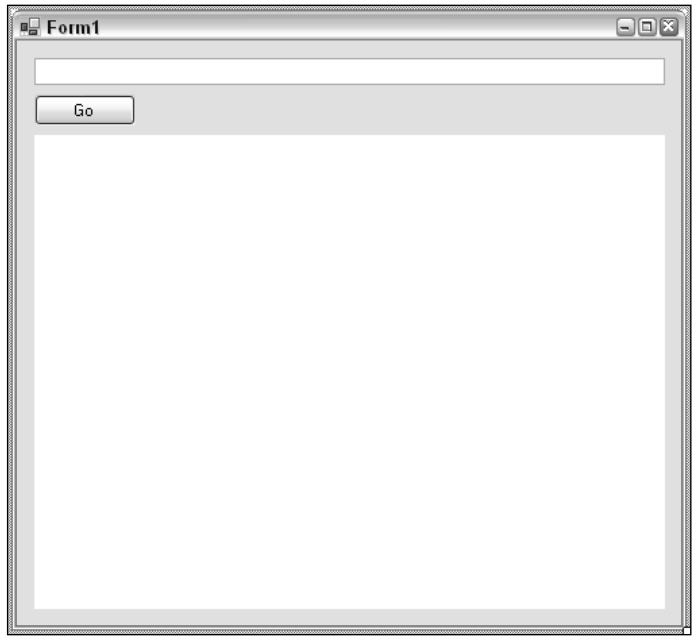

Figure 1-7

4. Now that you're done with design, all you need to do is add the code to make the button react when clicked. Just as you did in the previous Try It Out, double-click on the button to generate the code necessary to hook into the click of the button.

5. The only thing you need to do in the code is tell the `WebBrowser` control to go to the URL specified in the `TextBox` control. The properties you see in the Properties window are also accessible in code. The way you access these properties is by specifying the name of the control followed by a period (.) and then the name of the property. Methods are functions connected to an object, and they execute a certain task. In this case, you need the `Text` property of the `TextBox` control to get the URL text, and the `Navigate` method of the `WebBrowser` control to tell it to go to the URL. This is all achieved with the following line of code:

```
Private Sub Button1_Click(ByVal sender As System.Object, _
    ByVal e As System.EventArgs) Handles Button1.Click
        WebBrowser1.Navigate(TextBox1.Text)
End Sub
```

6. Run the program by pressing F5 or selecting the Start command on the Debug menu. When the form is displayed, type a URL in the text area, such as `http://www.wrox.com`, and then click the Go button. After a moment, the web browser area of the form will be populated with the web page associated with the URL, as shown in Figure 1-8.

Figure 1-8

As you can see, creating what appears to be a fairly complex program is made simple in Visual Basic 2005 Express. The controls used to create this program, along with the techniques known as *method* and *property access* in code, are discussed in the next few chapters.

Summary

Creating programs using Visual Basic 2005 Express is an immensely rewarding process. When you need to write only a couple of lines of code to achieve a feature-rich solution, it frees you to think of more complex solutions and helps you harness the power of Windows in ways that previously would be too difficult to contemplate.

In this chapter, you learned to do the following:

- Install Visual Basic Express, SQL Server Express, and the associated documentation
- Create a simple application that says "Hello World," and another that can browse a website

In the next chapter, you'll find out about starter kits and wizards — more features of Visual Basic Express that make your programming life easier. Along with these wizards, you'll learn about some core programming concepts such as controls, classes, methods, and properties that are essential to programming in any language.

Exercises

1. **Installing Visual Web Developer 2005 Express Edition:** To create applications that run on the Internet, you can still use Visual Basic 2005 as a language, but you will need to install Visual Web Developer 2005 Express Edition. The method for installing Web Developer Express is exactly the same as what has been outlined here, but it will install Web Developer instead of Visual Basic. If you have already installed Visual Basic Express, you'll find that the Web Developer installation process does not include options for MSDN or SQL Server, as it automatically detects that they are already present on your system.
2. **Customizing the Browser Application:** Extend your web browser program so you can both go back to the previous web page you visited and navigate to the default home page of Internet Explorer. You'll need to use two more methods of the `WebBrowser` control — `GoHome` and `GoBack`.